

## PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-134032

(43)Date of publication of application : 22.05.1998

(51)Int.Cl.

G06F 17/10

(21)Application number : 09-135922

(71)Applicant : SEIKO EPSON CORP

(22)Date of filing : 08.05.1997

(72)Inventor : KUBOTA SATORU  
MIYAYAMA YOSHIYUKI  
KUDO MAKOTO

(30)Priority

Priority number : 08234565

Priority date : 04.09.1996

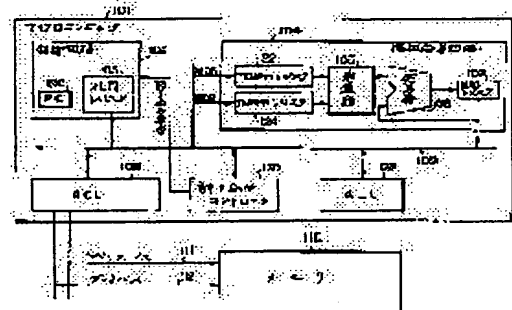
Priority country : JP

## (54) INFORMATION PROCESSING CIRCUIT, MICROCOMPUTER, AND ELECTRONIC EQUIPMENT

## (57)Abstract:

PROBLEM TO BE SOLVED: To improve memory use efficiency and an execution speed, to eliminate the critical path of product sum operation, and to prevent an overflow by performing product sum operation as many times as specified on the basis of execution frequency information included in a product sum operation instruction.

SOLUTION: A control circuit 102 performs control for receiving instructions including the product sum operation instruction, analyzing the received instructions, and executing the analyzed instructions. The product sum operation instruction contains the execution frequency information specifying the execution frequency of the product sum operation. Then a product sum arithmetic circuit 104 performs the product sum operation as many times as specified with the product sum operation instruction under the control of a control circuit 102. Consequently, the product sum operation can be performed as many times as desired with the one instruction. Therefore, the memory capacity needed for the product sum operation is greatly reduced and the use efficiency of a memory is improved. Further, the need to fetch the product sum operation instruction at each time during the execution of the product sum operation is eliminated to evade the delay of the product sum operation instruction execution.



## LEGAL STATUS

[Date of request for examination]  
[Date of sending the examiner's decision of rejection]  
[Kind of final disposal of application other than the examiner's decision of rejection or application converted r. gistration]  
[Date of final disposal for application]  
[Patent number]  
[Date of registration]  
[Number of appeal against examiner's decision of rejection]  
[Date of requesting appeal against examiner's decision of rejection]  
[Date of extinction of right]

Copyright (C); 1998,2000 Japanese Patent Office

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平10-134032

(43) 公開日 平成10年(1998) 5月22日

(51) Int.Cl.<sup>6</sup>

G 0 6 F 17/10

識別記号

F I

G 0 6 F 15/31

S

審査請求 未請求 請求項の数18 F D (全 21 頁)

(21) 出願番号 特願平9-135922

(22) 出願日 平成9年(1997) 5月8日

(31) 優先権主張番号 特願平8-234565

(32) 優先日 平8(1996) 9月4日

(33) 優先権主張国 日本 (J P)

(71) 出願人 000002369

セイコーエプソン株式会社

東京都新宿区西新宿2丁目4番1号

(72) 発明者 久保田 哲

長野県諏訪市大和3丁目3番5号 セイコーエプソン株式会社内

(72) 発明者 宮山 芳幸

長野県諏訪市大和3丁目3番5号 セイコーエプソン株式会社内

(72) 発明者 工藤 真

長野県諏訪市大和3丁目3番5号 セイコーエプソン株式会社内

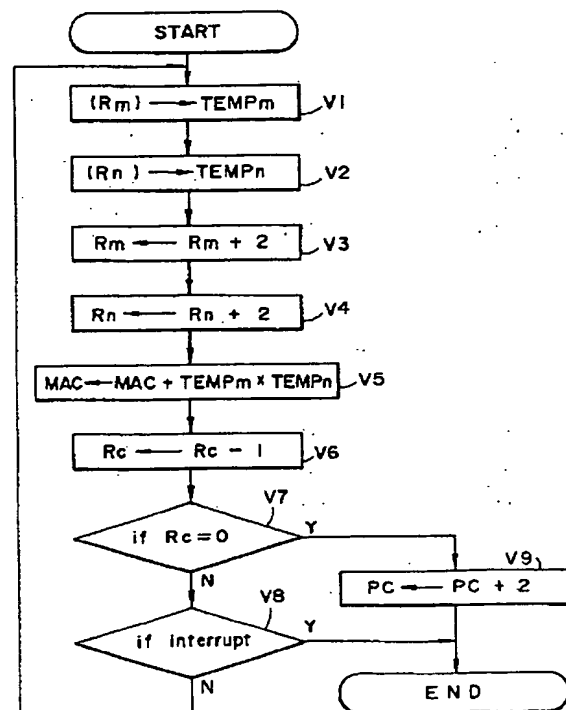
(74) 代理人 弁理士 井上 一 (外2名)

(54) 【発明の名称】 情報処理回路、マイクロコンピュータ及び電子機器

(57) 【要約】

【課題】 積和演算命令のメモリ使用効率、実行速度を改善し、積和演算におけるクリティカルパスの改善、オーバーフローの防止を図る。

【解決手段】 積和演算回路は、制御回路の制御の下で、積和演算命令に含まれる実行回数情報に基づき特定される回数だけ積和演算を実行する。積和演算の実行回数をレジスタに設定し、積和演算が1サイクル終了する毎にデクリメントし、レジスタの値が0になった時点で積和演算命令を終了する。複数回の積和演算の実行中に割り込みを受け付け、割り込み処理後に積和演算を継続実行する。1回のメモリアクセスで第1、第2の積和入力データを一度に読み出す。16ビット×16ビットの乗算結果を32ビットの加算器で加算し、下位の32ビットの加算でキャリー、ボローが生じた場合には、次のパイプラインステージで上位の32ビットのデータをインクリメント又はデクリメントする。



## 【特許請求の範囲】

【請求項1】 積和演算命令を含む命令を受け、該命令を解析し、該命令を実行するための制御を行う制御回路と、

前記積和演算命令に基づき前記制御回路の制御の下で積和演算を実行する積和演算回路とを含み、

前記積和演算回路が、

前記積和演算命令に含まれる実行回数情報に基づき特定される回数だけ積和演算を実行することを特徴とする情報処理回路。

【請求項2】 請求項1において、

前記制御回路が含むレジスタに格納される積和演算の実行回数を、積和演算の実行に同期させてデクリメントする回路を含み、

前記積和演算回路が、

前記実行回数が所与の値になるまで積和演算を実行することを特徴とする情報処理回路。

【請求項3】 請求項1又は2において、

前記積和演算命令が、

積和演算の実行回数用のレジスタ、第1の積和入力データ用のレジスタ及び第2の積和入力データ用のレジスタの中のいずれか1つのレジスタを指定するオペランドを含み、

前記制御回路が、

前記1つのレジスタ以外の他のレジスタを、前記1つのレジスタを指定する前記オペランドから所与のルールにしたがって特定することを特徴とする情報処理回路。

【請求項4】 請求項1又は2において、

前記積和演算命令が、

積和演算の実行回数用のレジスタを指定するオペランドと、第1の積和入力データ用のレジスタを指定するオペランドと、第2の積和入力データ用のレジスタを指定するオペランドとを含むことを特徴とする情報処理回路。

【請求項5】 請求項1又は2において、

前記制御回路が、

積和演算の実行回数に専用のレジスタ、第1の積和入力データに専用のレジスタ及び第2の積和入力データに専用のレジスタの少なくとも1つを含み、

前記積和演算命令が、

前記少なくとも1つの専用のレジスタを暗黙のオペランドとするオペレーションコードを含むことを特徴とする情報処理回路。

【請求項6】 請求項1乃至5のいずれかにおいて、

前記制御回路が、

前記実行回数情報に基づき特定される回数の積和演算の実行中に割り込み要求がなされた場合に該割り込み要求を受け付け、割り込み処理の終了後に、中断された積和演算を継続実行する制御を行うことを特徴とする情報処理回路。

【請求項7】 請求項6において、

前記制御回路が、

割り込み要求がなされた場合に、該制御回路が含むプログラムカウンタをインクリメントすることなく積和演算を一旦終了する制御を行い、

割り込み処理の終了後に、積和演算の実行回数用、第1の積和入力データ用及び第2の積和入力データ用のレジスタの割り込み処理分岐時点での内容に基づいて、積和演算を継続実行する制御を行うことを特徴とする情報処理回路。

10 【請求項8】 請求項6又は7において、

前記実行回数情報に基づき特定される回数だけ積和演算を実行した場合及び前記割り込み要求がなされた場合のいずれかの場合にアクティブになる信号に基づいて、積和演算回路のステートを初期ステートに戻すステートマシーンを含むことを特徴とする情報処理回路。

【請求項9】 請求項6乃至8のいずれかにおいて、

第1の積和入力データ用のレジスタの内容及び第2の積和入力データ用のレジスタの内容が、積和演算の継続実行の際に使用される内容に変化した後に、積和演算回路のステートを初期ステートに戻すステートマシーンを含むことを特徴とする情報処理回路。

【請求項10】 請求項1乃至9のいずれかにおいて、

前記制御回路が、

第1、第2の積和入力データが隣り合って格納されるメモリ上の領域から、該第1、第2の積和入力データを1回のメモリアクセスで読み出す制御を行うことを特徴とする情報処理回路。

【請求項11】 積和演算命令を含む命令を受け、該命令を解析し、該命令を実行するための制御を行う制御回路と、

前記積和演算命令に基づき前記制御回路の制御の下で積和演算を実行する積和演算回路とを含み、

前記制御回路が、

第1、第2の積和入力データが隣り合って格納されるメモリ上の領域から、該第1、第2の積和入力データを1回のメモリアクセスで読み出す制御を行うことを特徴とする情報処理回路。

【請求項12】 請求項10又は11において、

前記制御回路と前記メモリとの間でのデータ転送が $2^n$ ビットのバスで行われる場合に、転送されるデータの上位 $2^{n-1}$ ビットを前記第1の積和入力データとし、下位 $2^{n-1}$ ビットを前記第2の積和入力データとすることを特徴とする情報処理回路。

【請求項13】 請求項1乃至12のいずれかにおいて、

前記積和演算回路が、

パイプライン処理の第1のステージにおいて、

第1、第2の積和入力データを乗算し、

パイプライン処理の第2のステージにおいて、

50 所与の第1の積和結果用レジスタに格納されるデータに

## 3

前記第1のステージの乗算結果を加算し、  
パイプライン処理の第3のステージにおいて、  
前記第2のステージの加算でキャリー及びボローのいずれかが生じた場合に、所与の第2の積和結果用レジスタに格納されるデータのインクリメント及びデクリメントのいずれかを行うことを特徴とする情報処理回路。

【請求項14】 積和演算命令を含む命令を受け、該命令を解析し、該命令を実行するための制御を行う制御回路と、

前記積和演算命令に基づき前記制御回路の制御の下で積和演算を実行する積和演算回路とを含み、

前記積和演算回路が、

パイプライン処理の第1のステージにおいて、

第1、第2の積和入力データを乗算し、

パイプライン処理の第2のステージにおいて、

所与の第1の積和結果用レジスタに格納されるデータに前記第1のステージの乗算結果を加算し、

パイプライン処理の第3のステージにおいて、

前記第2のステージの加算でキャリー及びボローのいずれかが生じた場合に、所与の第2の積和結果用レジスタに格納されるデータのインクリメント及びデクリメントのいずれかを行うことを特徴とする情報処理回路。

【請求項15】 請求項13又は14において、

前記積和演算回路が、

前記第1、第2の積和結果用レジスタと、

前記第1、第2の積和入力データを乗算する乗算器と、

前記第1の積和結果用レジスタに格納されるデータに前記乗算器からの乗算結果を加算する加算器と、

前記加算器からのキャリー信号、ボロー信号に基づいて、前記第2の積和結果用レジスタに格納されるデータのインクリメント及びデクリメントのいずれかを行う回路とを含むことを特徴とする情報処理回路。

【請求項16】 請求項13乃至15のいずれかにおいて、

前記第1、第2の積和入力データの各々が $2^{n-1}$ ビットのデータであり、前記第1、第2の積和結果用レジスタの各々が $2^n$ ビットのレジスタであることを特徴とする情報処理回路。

【請求項17】 半導体基板上に集積されたマイクロコンピュータであって、

請求項1乃至16のいずれかの情報処理回路と、

バスコントロール回路、メモリ、割り込みコントローラ、タイマ回路、アナログインターフェース回路、データ転送制御回路及びI/O回路の少なくとも1つとを含むことを特徴とするマイクロコンピュータ。

【請求項18】 請求項17のマイクロコンピュータと、

前記マイクロコンピュータの処理対象となるデータの入力源と、

前記マイクロコンピュータにより処理されたデータを出

## 4

力するための出力装置とを含むことを特徴とする電子機器。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、情報処理回路、マイクロコンピュータ及び電子機器に関する。

【0002】

【背景技術及び発明が解決しようとする課題】近年、積和演算命令を高速に実行できるマイクロコンピュータに対する需要が高まっている。高速な積和演算が可能になると、これまでDSP (Digital Signal Processor)、画像処理専用IC、音処理専用ICが行っていた処理をマイクロコンピュータに代行させることが可能となり、製品の低コスト化、システムの簡素化を図れるからである。

【0003】マイクロコンピュータにおける、積和演算命令は例えば次のように実行される。まずメモリ上の第1の領域に第1の積和入力データを格納しておくと共に第2の領域に第2の積和入力データを格納しておく。次にマイクロコンピュータが内蔵する汎用レジスタの内容で指定される2個のアドレスを用いて、上記第1、第2の領域に格納された第1、第2の積和入力データをメモリから読み出す。そしてこれらの第1、第2の積和入力データを乗算し、その結果を積和演算回路が内蔵する積和結果用のレジスタ (MACレジスタ) に加算する動作を行う。

【0004】しかしながら、積和演算命令の実行が可能な上記マイクロコンピュータには次のような課題がある。

【0005】(1) 積和演算を複数回実行しようとした場合、積和演算命令を繰り返す回数分だけ、積和演算の命令を並べたプログラムを作成する必要がある。そのため、積和演算の実行回数が増大すると、それに応じて積和演算命令を格納するのに必要なメモリ容量が増加してしまう。これを解決する1つの手法として、積和演算の実行回数をデクリメントしながら積和演算を実行し、実行回数が零になった時にループを抜けるようなプログラムを作成する手法も考えられる。しかしながら、この手法によると、積和演算の1回当たりの実行に要する時間が長くなる。

【0006】(2) 積和演算を連続して実行した場合、1回当たりの実行時間が、第1、第2の積和入力データをメモリから読み出す時間によって制限されてしまう。

【0007】(3) これまでは例えば各々が16ビットの第1、第2の積和入力データを乗算し、乗算結果を48ビットのMACレジスタ (積和結果用レジスタ) に加算していた。この場合、48ビットの加算を1クロック内で完了する必要がある、この加算処理がクリティカルパスとなる。またMACレジスタが48ビットのビット長しか持たないため、積和演算の実行回数が多くなる

## 5

と、すぐにオーバーフローしてしまう。

【0008】本発明は、以上のような技術的課題に鑑みてなされたものであり、その目的とするところは、積和演算命令を用いるプログラムのメモリ使用効率の向上を図れる情報処理回路、マイクロコンピュータ及び電子機器を提供することにある。

【0009】また本発明の他の目的は、積和演算命令の実行速度を改善できる情報処理回路等を提供することにある。

【0010】また本発明の他の目的は、積和演算におけるクリティカルパスの解消、積和演算におけるオーバーフローの防止を図れる情報処理回路等を提供することにある。

## 【0011】

【課題を解決するための手段】上記課題を解決するために本発明に係る情報処理回路は、積和演算命令を含む命令を受け、該命令を解析し、該命令を実行するための制御を行う制御回路と、前記積和演算命令に基づき前記制御回路の制御の下で積和演算を実行する積和演算回路とを含み、前記積和演算回路が、前記積和演算命令に含まれる実行回数情報に基づき特定される回数だけ積和演算を実行することを特徴とする。

【0012】本発明によれば、積和演算命令に、積和演算の実行回数を特定するための実行回数情報が含まれる。そして積和演算回路は、制御回路の制御の下で、積和演算命令により特定された回数の積和演算を行う。これにより、1回の命令で所望の回数の積和演算を実行することが可能となる。従って、積和演算の回数だけ積和演算命令を並べる手法に比べて、積和演算に必要とされるメモリ容量を大幅に削減でき、メモリの使用効率を向上できる。また積和演算を実行中に積和演算命令を毎回フェッチする必要がなくなり、積和演算命令実行の遅延を回避できる。

【0013】また本発明は、前記制御回路が含むレジスタに格納される積和演算の実行回数を、積和演算の実行に同期させてデクリメントする回路を含み、前記積和演算回路が、前記実行回数が所与の値になるまで積和演算を実行することを特徴とする。このようにすれば、積和演算の実行毎に実行回数をメモリから読み出す必要がなくなり、処理速度を向上できる。またこのようにすることで、複数回の積和演算の実行中に割り込みが発生した場合に、割り込み処理後に、レジスタに格納されている実行回数に基づいて積和演算を継続実行することが可能となる。

【0014】なお本発明では、前記積和演算命令が、積和演算の実行回数用のレジスタ、第1の積和入力データ用のレジスタ及び第2の積和入力データ用のレジスタの中のいずれか1つのレジスタを指定するオペランドを含み、前記制御回路が、前記1つのレジスタ以外の他のレジスタを、前記1つのレジスタを指定する前記オペラン

## 6

ドから所与のルールにしたがって特定することが望ましい。このようにすれば、命令のビット長を小さくすることができ、プログラムのコードサイズの縮小化を図れる。

【0015】なお前記積和演算命令が、積和演算の実行回数用のレジスタを指定するオペランドと、第1の積和入力データ用のレジスタを指定するオペランドと、第2の積和入力データ用のレジスタを指定するオペランドとを含むようにしてもよい。また前記制御回路が、積和演算の実行回数に専用のレジスタ、第1の積和入力データに専用のレジスタ及び第2の積和入力データに専用のレジスタの少なくとも1つを含み、前記積和演算命令が、前記少なくとも1つの専用のレジスタを暗黙のオペランドとするオペレーションコードを含むようにしてもよい。

【0016】また本発明は、前記制御回路が、前記実行回数情報に基づき特定される回数の積和演算の実行中に割り込み要求がなされた場合に該割り込み要求を受け付け、割り込み処理の終了後に、中断された積和演算を継続実行する制御を行うことを特徴とする。このようにすれば、複数回の積和演算の実行の際に、割り込みが長時間待たされるという問題を解消できる。

【0017】また本発明は、前記制御回路が、割り込み要求がなされた場合に、該制御回路が含むプログラムカウンタをインクリメントすることなく積和演算を一旦終了する制御を行い、割り込み処理の終了後に、積和演算の実行回数用、第1の積和入力データ用及び第2の積和入力データ用のレジスタの割り込み処理分岐時点での内容に基づいて、積和演算を継続実行する制御を行うことを特徴とする。このようにすれば、割り込み処理の終了後に、適正な第1、第2の積和入力データを用いて、適正な実行回数だけ積和演算を継続実行することが可能となる。

【0018】また本発明は、前記実行回数情報に基づき特定される回数だけ積和演算を実行した場合及び前記割り込み要求がなされた場合のいずれかの場合にアクティブになる信号に基づいて、積和演算回路のステートを初期ステータスに戻すステートマシンの含むことを特徴とする。このようにすれば、ステートマシンの構成の簡素化を図れる。

【0019】また本発明は、第1の積和入力データ用のレジスタの内容及び第2の積和入力データ用のレジスタの内容が、積和演算の継続実行の際に使用される内容に変化した後に、積和演算回路のステートを初期ステータスに戻すステートマシンの含むことを特徴とする。このようにすれば、割り込み処理の終了後に、適正な第1、第2の積和入力データを用いて積和演算を継続実行することが可能となる。

【0020】また本発明は、前記制御回路が、第1、第2の積和入力データが隣り合って格納されるメモリ上の

領域から、該第1、第2の積和入力データを1回のメモリアクセスで読み出す制御を行うことを特徴とする。これまでは、積和演算の1回当たりの実行時間は、2回のメモリアクセスに要する時間に制限されていたが、本発明によれば、上記実行時間を、1回のメモリアクセスに要する時間にまで短縮できる

また本発明は、前記制御回路と前記メモリとの間でのデータ転送が $2^n$ ビットのバスで行われる場合に、転送されるデータの上位 $2^{n-1}$ ビットを前記第1の積和入力データとし、下位 $2^{n-1}$ ビットを前記第2の積和入力データとすることを特徴とする。このようにすることで、第1、第2の積和入力データを $2^n$ ビットのバスを用いて1回のメモリアクセスで読み込むことが可能となる。

【0021】また本発明は、前記積和演算回路が、パイプライン処理の第1のステージにおいて、第1、第2の積和入力データを乗算し、パイプライン処理の第2のステージにおいて、所与の第1の積和結果用レジスタに格納されるデータに前記第1のステージの乗算結果を加算し、パイプライン処理の第3のステージにおいて、前記第2のステージの加算でキャリー及びボローのいずれかが生じた場合に、所与の第2の積和結果用レジスタに格納されるデータのインクリメント及びデクリメントのいずれかを行うことを特徴とする。このようにすれば、乗算結果を加算する際に生じるクリティカルパスの問題、オーバーフローの問題を解消できる。

【0022】なお本発明は、前記積和演算回路が、前記第1、第2の積和結果用レジスタと、前記第1、第2の積和入力データを乗算する乗算器と、前記第1の積和結果用レジスタに格納されるデータに前記乗算器からの乗算結果を加算する加算器と、前記加算器からのキャリー信号、ボロー信号に基づいて、前記第2の積和結果用レジスタに格納されるデータのインクリメント及びデクリメントのいずれかを行う回路とを含むことが望ましい。

【0023】また本発明は、前記第1、第2の積和入力データの各々が $2^{n-1}$ ビットのデータであり、前記第1、第2の積和結果用レジスタの各々が $2^n$ ビットのレジスタであることを特徴とする。このようにすれば、 $2^n$ ビットの第1の積和結果用レジスタと $2^n$ ビットの第2の積和結果用レジスタとを合わせて $2^{n+1}$ ビットのレジスタに対して積和演算結果を加算できるため、積和演算の実行可能回数を、事実上、無限大にすることが可能となる。

【0024】また本発明は、半導体基板上に集積されたマイクロコンピュータであって、上記のいずれかの情報処理回路と、バスコントロール回路、メモリ、割り込みコントローラ、タイマ回路、アナログインターフェース回路、データ転送制御回路及びI/O回路の少なくとも1つを含むことを特徴とする。このようなマイクロコンピュータを用いることで、これまでDSP、画像処理専用IC或いは音処理専用ICが行っていた処理を代行

することが可能となる。

【0025】また本発明に係る電子機器は、上記のマイクロコンピュータと、前記マイクロコンピュータの処理対象となるデータの入力源と、前記マイクロコンピュータにより処理されたデータを出力するための出力装置とを含むことを特徴とする。このようにすれば、これまでDSP等が行っていた画像処理、音処理をマイクロコンピュータが代行することが可能となり、電子機器の低コスト化、コンパクト化、低消費電力化を図ることが可能となる。

【0026】

【発明の実施の形態】以下、本発明の好適な実施形態について図面を用いて詳細に説明する。なお以下では、本発明の情報処理回路をマイクロコンピュータに適用した場合を主に例にとり説明を行う。

【0027】（実施例1）実施例1は、積和演算命令に含まれる実行回数情報に基づき特定される回数だけ積和演算回路に積和演算を実行させる実施例である。

【0028】図1に、積和演算回路104を内蔵するマイクロコンピュータ101のブロック図を示す。この機能ブロック図は、実施例1及び後述の実施例2、3、4に共通のブロック図である。図1では、マイクロコンピュータ101は32ビットのデータを処理する。また積和演算回路104は、各々が16ビットの第1、第2の積和入力データMDA、MDBを乗算し、その乗算結果である32ビットのデータを64ビットのMACレジスタ107に加算する。但し本発明の適用範囲は、マイクロコンピュータ及び積和演算回路の処理するデータのビット長、もしくはマイクロコンピュータの内蔵する汎用レジスタの数等により限定されるものではない。

【0029】図1のマイクロコンピュータ101は、32ビットのデータを処理する制御回路102と、マイクロコンピュータ101とメモリ110とを接続するバスを制御するバスコントロールユニットBCU108と、積和演算を実行する積和演算回路104と、マイクロコンピュータの内部や外部からの種々の割り込みを受け付け制御回路102に割り込み要求を出す割り込みコントローラ130と、データの加減演算などの算術演算及び論理和、論理積、論理シフトなどの論理演算を行うALU（算術論理演算ユニット）132を含む。

【0030】ここで制御回路102は、積和演算命令を含む命令を受け、受け付けた命令を解析し、解析した命令を実行するための制御を行うものであり、16ビット長の命令を使用している。また制御回路102は、R0からR15までの16本の32ビットのレジスタから成る汎用レジスタ103と、PC（プログラムカウンタ）120とを含む。この制御回路102の制御の下で、積和演算回路104は積和演算を実行し、ALU132は算術演算、論理演算を実行する。そしてこれらの制御回路102、積和演算回路104及びALU132が、マ

マイクロコンピュータ101の中央処理ユニット(CPU)として機能する。

【0031】制御回路102、BCU108及び積和演算回路104は、内部データバス109を介してデータ転送を行う。BCU108は、外部アドレスバス111と外部データバス112を用いて、メモリ110から第1、第2の積和入力データMDA、MDBを読み込む。但し本発明の適用範囲は、メモリ110がマイクロコンピュータ101に内蔵されるか、マイクロコンピュータ101の外部に設けられるかによって限定されるものではない。

【0032】積和演算回路104は、第1、第2の積和入力データMDA、MDBを一時的に保持するTEMPmレジスタ122及びTEMPnレジスタ124と、一時的に保持されたMDA、MDBを乗算する乗算器105と、この乗算結果を用いて加算処理を行う加算器106と、加算結果を保持する64ビットのMACレジスタ(積和結果用レジスタ)107を含む。積和演算回路104は、16ビットのMDA、MDBを入力として、その乗算結果とMACレジスタ107の内容とを加算し、その加算結果をMACレジスタ107に格納する動作を行う。

【0033】次に図2のフローチャート及び図3

(A)、(B)、(C)を用いて本実施例の動作について説明する。

【0034】まず積和演算命令実行前に、図3(A)に示すように、メモリ上の第1の領域10に第1の積和入力データMDA0~MDA1を格納し、第2のメモリ領域12に第2の積和入力データMDB0~MDB1を格納しておく。また汎用レジスタ103が含むレジスタRm、Rnに第1、第2の領域10、12の先頭アドレスをロードしておき、第1、第2の積和入力データの先頭データMDA0、MDB0をRm、Rnが指すようにしておく。また汎用レジスタ103が含むレジスタRcに、積和演算の実行回数をロードしておくと共に、MACレジスタ107を初期化しておく。

【0035】この状態で制御回路102が積和演算命令を受け付けると(即ち図3(A)に示すようにPC120の指す命令が積和演算命令であると)、この積和演算命令を解析する制御回路102の制御の下で、積和演算命令実行のための種々の処理が行われる。

【0036】即ちまず、レジスタRmに格納されるアドレスにより指定される第1の積和入力データMDA0を、BCU108を介してメモリ110から読み出し、TEMPmレジスタ122に格納する(図2のステップS1)。同様にRnが指すMDB0を、BCU108を介してメモリ110から読み出し、TEMPnレジスタ124に格納する(ステップS2)。なお、ステップS1、S2の(Rm)、(Rn)は、各々、Rm、Rnに格納されるアドレスにより指定されるメモリ上のデータ

(第1、第2の積和入力データ)を意味する。

【0037】次に、Rm、Rnに格納されるアドレスを+2(番地)する(ステップS3、S4)。本実施例では、メモリ空間は、8ビットの1バイトを最小単位としてアドレッシングされており、第1、第2の積和入力データは16ビットのデータである。従ってRm、Rnのアドレスを+2すると、図3(B)に示すようにRm、Rnは次の積和入力データMDA1、MDB1を指すようになる。

10 【0038】次にTEMPmレジスタ122、TEMPnレジスタ124に格納されている16ビットのMDA0、MDB0を乗算器105を用いて乗算し、その乗算結果とMACレジスタ107の内容を加算器106により加算し、その加算結果をMACレジスタ107に格納する(ステップS5)。

【0039】次に、レジスタRcに格納される積和演算の実行回数をデクリメントする(ステップS6)。即ち積和演算の実行回数を、積和演算に同期させてデクリメントする。なお図2では積和演算が終了する毎に実行回数をデクリメントしているが、実行回数のデクリメントは少なくとも積和演算の実行に同期するものであればよい。このデクリメント処理は、例えば図1のALU132が行う。

【0040】次に、Rcに格納される積和演算の実行回数が所与の値、例えば零か否かを判断する(ステップS7)。零でない場合には、ステップS1に戻る。この時、前述のようにステップS3、S4でRm、Rnが+2(番地)されているため、積和演算のオペランドは次の積和入力データMDA1、MDB1となっている(図3(B)参照)。

【0041】一方、Rcに設定された実行回数だけ積和演算が繰り返され、Rcの値が零になると、図3(C)に示すようにPC120に格納されるアドレスを+2(番地)する(ステップS8)。これにより積和演算命令が終了する共にPC120が次の命令を指すようになる。本実施例では、命令長は2バイトの16ビットであるため、PC120の値を+2(番地)すると、2バイト先の次の命令が指されることになる。

【0042】以上のような処理が行われるように制御回路102が、積和演算回路104、BCU108、ALU132などを制御することで、1つの命令で所望の回数だけ積和演算を実行することが可能となる。

【0043】図4に、本実施例の比較例となるマイクロコンピュータの処理のフローチャートを示す。図4のステップT1~T5は図2のステップS1~S5と同様である。しかしながら、図4の比較例では図2の本実施例と異なり、Rcの値をデクリメントしたり、Rcの値が零か否かを判断することなく、PC120の値を+2して積和演算命令を終了する。また積和演算命令に、積和演算の実行回数を特定するための情報が含まれていない。



【0044】従って図4の手法で積和演算を所望の回数だけ実行するためには、図5(A)に示すように、その回数分だけ積和演算の命令を並べたプログラムを作成する必要があり、これはプログラムのコードサイズの増大化、命令を格納するメモリの大規模化等の問題を招く。

【0045】このような問題を解決する1つの手法として、図5(B)に示すように、積和演算を実行する回数Cをデクリメントしながら積和演算を実行し(ステップU1、U2)、実行回数Cが零になった場合にループを抜ける(ステップU3)ようにプログラムする手法も考えられる。しかしながら、この手法によると、積和演算の1回当たりの実行に要する時間が長くなる。即ち、この手法によると、積和演算命令(ステップU1)の実行に要する時間以外に、実行回数Cをデクリメントする命令(ステップU2)に要する時間、実行回数が零か否かを判断する命令(ステップU3)に要する時間が余計になる。従って、図2の本実施例に比べ、最低でも2クロック分だけ処理時間が増えてしまい、処理速度の低下を招く。

【0046】これに対して本実施例では、実行回数分だけ積和演算命令を並べるプログラムや、実行回数をデクリメントし実行回数が零か否かを判断するプログラムを作成する必要がないため、コードサイズの効率化、命令を格納するメモリの小規模化、処理の高速化を図りながら、1つの積和演算命令で所望の回数だけ積和演算命令を実行することが可能となる。

【0047】また本実施例によれば、積和演算を実行中に積和演算命令を毎回フェッチする必要がない。従って、積和入力データの読み込みと積和演算命令のフェッチが競合することで発生する積和演算命令実行の遅延を回避できると共に、無駄なフェッチによる電力消費を削減できる。

【0048】さて積和演算の実行回数を特定させる積和演算命令の実施形態としては種々のものが考えられる。

【0049】例えば図6(A)では、積和演算命令に、複数の命令の中から積和演算命令を指定するための6ビットのオペレーションコードと、レジスタRcを16個の汎用レジスタの中から指定する4ビットのオペランドとを含ませている。そしてこの場合には、制御回路102が、第1の積和入力データ用のレジスタRm及び第2の積和入力データ用のレジスタRnを、上記Rcを指定するオペランドに基づき所与のルールにしたがって特定する。例えば図6(A)の積和演算命令で汎用レジスタR13がRcに割り当てられた場合には、R13を+1、+2した汎用レジスタR14、R15に、各々、Rm、Rnを割り当てる。このようにすることで、命令長を16ビット以内に抑えることが可能となり、コードサイズの効率化、命令を格納するメモリの小規模化を図れる。特に全ての命令の長さを例えば16ビットの固定長にしてコードサイズの効率化を図る命令セットのアーキ

テクチャにおいては、図6(A)の手法は有効である。なお図6(A)では、積和演算命令が、Rcを指定するオペランドを含んでいるが、Rm或いはRnを指定するオペランドを含ませるようにしてもよい。

【0050】また図6(B)では、積和演算命令に、6ビットのオペレーションコード、Rcを指定する4ビットのオペランド、Rmを指定する4ビットのオペランド及びRnを指定する4ビットのオペランドを含ませている。即ち積和演算命令によりRc、Rm、Rnを直接指定している。この手法は図6(A)に比べて、命令長が長くなるという不利点があるが、1つのオペランドから他のオペランドを特定する処理が必要がないという利点がある。

【0051】また図6(C)では、Rc、Rm、Rnを、各々、実行回数、第1、第2の積和入力データに専用のレジスタにしている。この場合には、積和演算命令は、これらの専用のレジスタを暗黙のオペランドとするオペレーションコードを含むことになる。このようにすることでコードサイズの効率化を図れる。なお図6

(C)では、Rc、Rm、Rnを全て専用レジスタにしているが、これらの中の1つ或いは2つのみを専用レジスタにするようにしてもよい。

【0052】なお本実施例では、第1、第2の積和入力データについてはレジスタRm、Rnに格納されるアドレスに基づいて積和演算の実行毎にメモリから読み出しているのに対して、実行回数についてはRc自体に格納されているものを使用している。積和演算の実行毎に実行回数をメモリから読み出すと、処理速度の低下を招くからである。また本実施例では、実行回数を、一度メモリからレジスタRcにロードした後にALU132を用いてデクリメントしているため、積和入力データの場合同様に再度メモリから実行回数を読み出す必要性がないからである。但し本発明の範囲は、このような実行回数の指定手法に限定されるものではない。

【0053】(実施例2) 実施例2は、複数回の積和演算の実行中に割り込みを受け付けると共に、割り込み終了後に、中断された積和演算を継続実行する実施例である。以下、図7のフローチャート及び図8(A)、

(B)を用いて実施例2について説明する。

【0054】前述の実施例1では、1つの積和演算命令で複数回の積和演算を実行することができる。しかしながら、この複数回の積和演算の実行は1つの命令と見なされるため、Rcに設定された実行回数分の積和演算が完了しない限り、PC120の値は変化せず、次の命令に移行しない。一方、図1の割り込みコントローラ130からの割り込み要求は、通常は、命令と命令の境界で処理される。従って複数回の積和演算を実行している期間、割り込み処理への移行が長時間待たされるという不具合が生じる。

【0055】そこで実施例2では、図7に示すような処

理を行っている（なお図7のステップV1～V6は図2のステップS1～S6と同様である）。即ちRcの値である積和演算の実行回数が零か否かの判断（ステップV7）の後に、割り込み要求が存在するか否かを判断する（ステップV8）。そして割り込み要求が存在しない場合は、積和演算の処理を継続し、次の積和入力データに基づく積和演算を行う。一方、割り込み要求が存在する場合は、PC120をインクリメントすることなく（ステップV9の処理を省略する）、積和演算命令を一旦終了する。図8（A）では、例えば積和入力データMDAK-1、MDBK-1に対する積和演算処理の際に割り込み要求が生じている。この場合、図7のステップV3、V4の処理により、Rm、Rnは次の積和入力データMDAK、MDBKを指すようになっている。またPC120は、次の命令を指さず積和演算命令を指したままとなっている（図8（A）のH1参照）。

【0056】積和演算命令を終了することで、割り込み処理プログラムに分岐して、割り込み処理を実行することが可能となる。割り込み処理は、通常、命令と命令の境界で行われるからである。ここで割り込み処理のハンドラは、割り込み処理に分岐する前に、割り込み処理が終了した時の戻りアドレスを指すPC120の値をスタックする。しかしながら図7のステップV8、V9に示すように本実施例では、積和演算命令の実行中に割り込み処理に分岐した場合には、PC120の値を+2することなく積和演算命令を終了している。従って、図8（B）のH2に示すようにPC120は積和演算命令を指したままとなっているため、割り込み処理の終了後に、再度同じ積和演算命令が実行されることになる。

【0057】そして、この時、図8（B）のH3、H4、H5に示すように、Rc、Rm及びRnの値は、割り込み処理に分岐する時点での値になっている。従って、割り込み処理で中断された積和演算を適正に継続実行することが可能となる。即ち、割り込み処理による中断前に、積和入力データMDAK-1、MDBK-1までの処理が完了しており、中断後に、MDAK、MDBKに対する処理から積和演算を再実行できるようになる。

【0058】以上説明した実施例2によれば、複数回の積和演算の実行中に割り込みを受け付け割り込み処理を行うことができ、しかも、割り込み処理の待ち時間も、図4の場合の待ち時間と同等にすることができる。

【0059】（実施例3）実施例3は、第1、第2の積和入力データが隣り合って格納されるメモリ上の領域から、これらの第1、第2の積和入力データを1回のメモリアクセスで読み出す実施例である。以下、図9のフローチャート及び図10（A）、（B）、（C）を用いて実施例3について説明する。

【0060】実施例3では、図10（A）に示すように、第1の積和入力データMDAと第2の積和入力データMDBとを隣り合わせてメモリ上に格納する。例えば

MDA0の次にはMDB0が格納され、MDB0の次にはMDA1、MDB1が格納される。即ちNを自然数として、第1の積和入力データMDAは4N番地のアドレスに、第2の積和入力データMDBは4N+2番地のアドレスに格納される。この点において、第1の領域10に第1の積和入力データMDAがひとまとめに格納され、第2の領域12に第2の積和入力データMDBがひとまとめに格納される図3（A）、（B）、（C）と相違する。

10 【0061】そして実施例3では、実施例1、2と異なり、3個ではなく2個のレジスタRm、Rcを使用する。即ちRmによりMDA0、MDB0を読み出し、Rmの値を+4することによりMDA1、MDB1を読み出すようにする。

【0062】実施例3の動作について説明する。まずレジスタRmに格納されるアドレスにより指定される第1、第2の積和入力データMDA0、MDB0をメモリから読み出し、読み出されたデータの上位16ビット（2バイト）をTEMPmレジスタ122に、下位16ビットをTEMPnレジスタ124に格納する（ステップW1）。

20 【0063】即ち本実施例では、図10（A）に示すように、メモリ110と積和演算回路104の間のデータ転送は32ビットのバスにより行われる。従って、1回のメモリアクセスで、32ビット（4バイト）のデータを読み込むことができ、その読み出しデータの上位16ビットが第1の積和入力データMDA0となり、下位16ビットが第2の積和入力データMDB0になる。

【0064】積和入力データを読み出した後、Rmの値を+4し、積和演算を実行する（ステップW2、W3）。Rmの値を+4することにより、図10（A）に示すように次の積和入力データMDA1、MDB1を指定することが可能となる。なおステップW6～W9の処理は、図7のステップV6～V9と同様である。

【0065】実施例1、2では、図10（B）に示すように、1回のメモリアクセス（1クロック期間）で、第1、第2の積和入力データMDA、MDBのいずれか一方しか読み出せない。そして積和演算はMDA、MDBの両方が揃わないと実行できないため、結局、積和演算を2クロック毎にしか実行できない。即ち積和演算の1回当たりの実行時間は、2回のメモリアクセス分の時間となっていた。

【0066】これに対して実施例3では、図10（C）に示すように、1回のメモリアクセスで第1、第2の積和入力データMDA、MDBの両方を読み出すことができる。従って、積和演算を1クロック毎に行うことが可能となり、積和演算の実行時間を1回のメモリアクセス分の時間とすることが可能となる。これにより、処理速度を格段に向上できる。

50 【0067】（実施例4）実施例4は、パイプライン方

式の積和演算において、第1のステージで乗算を行い、第2のステージで乗算結果を下位の第1の積和結果用レジスタに加算し、第1の積和結果用レジスタがオーバーフローした場合に、第3のステージで上位の第2の積和結果用レジスタをインクリメント又はデクリメントする実施例である。

【0068】図11(A)に、実施例4の積和演算回路のブロック図を示す。この積和演算回路は、乗算器105、加算器106-1、インクリメント・デクリメンタ106-2、MACレジスタ(積和結果用レジスタ)の下位のレジスタであるALR107-1、MACレジスタの上位のレジスタであるAHR107-2を含む。

【0069】図11(B)に示すように、パイプライン処理の第1のステージでは、乗算器105が第1、第2の積和入力データMDA、MDBを乗算する。次にパイプライン処理の第2のステージにおいて、加算器106-1が、ALR107-1(第1の積和結果用レジスタ)に格納されるデータに第1のステージの乗算結果を加算する。そして第2のステージの加算で正のオーバーフローが生じキャリー信号がアクティブになった場合には、パイプライン処理の第3のステージにおいて、インクリメント・デクリメンタ106-2がAHR107-2(第2の積和結果用レジスタ)に格納されるデータをインクリメントする。一方、第2のステージの加算で負のオーバーフローが生じ、ボロー信号がアクティブになった場合には、第3のステージにおいて、インクリメント・デクリメンタ106-2がAHR107-2に格納されるデータをデクリメントする。

【0070】このように実施例4では、16ビット×16ビットの乗算で得られた結果を、ALR107-1、AHR107-2から成る64ビットのMACレジスタに加算する。そしてこの加算を、下位の32ビットの加算と上位の32ビットの加算に分け、下位の32ビットの加算をパイプライン処理の第2のステージで実行し、上位の32ビットの加算(インクリメント又はデクリメント)をパイプライン処理の第3のステージで実行する。これにより以下の効果を得ることができる。

【0071】加算器106-1を48ビットではなく32ビットにすることができ、48ビットの加算器を使用する場合に問題となっていたクリティカルパスを解消できる。

【0072】MACレジスタ(ALR107-1、AHR107-2)のビット数を64ビットに拡張できるため、積和演算の際のオーバーフロー(飽和)の可能性を減らすことができると共に、積和演算の実行回数を2<sup>32</sup>-1回にすることができ、事実上、無制限にすることができる。特に実施例4は実施例1との組み合わせにおいて特有の効果を奏する。即ち積和演算命令に実行回数を特定する情報を含ませる実施例1の手法によれば、積和演算命令を実行回数分だけ並べたプログラムを作成する

必要がないため、ユーザが指定する積和演算の実行回数が非常に大きくなる可能性がある。実施例4によれば、積和演算命令の実行回数を事実上無制限にできるため、このような大きな実行回数の指定に対しても対処できる。

【0073】通常の加算器に比べてハードウェア規模が小さいインクリメント・デクリメンタ106-2を用いて上位32ビットの加算処理ができる。従ってMACレジスタ(ALR107-1、AHR107-2)のビット数を64ビットに拡張したにも関わらず、ハードウェアの大規模化を最小限に抑えることができる。

【0074】なお1つのハードウェアを用いて符号付きのデータの乗算と符号なしのデータの乗算を扱えるようにするためには、乗算器105を17ビット×17ビットの構成とすることが望ましい。また図11(A)のキャリー信号は、加算器106-1がキャリーを発生し且つその時のデータが正の場合にアクティブになる。一方、ボロー信号は、加算器106-1がキャリーを発生し且つその時のデータが負の場合にアクティブになる。また符号なしのデータのみを扱う場合には、ボロー信号は必要なく、インクリメント・デクリメンタ106-2はインクリメントの機能を有するのみでよい。

【0075】(実施例5) 実施例5は、図1の制御回路102、積和演算回路104、ALU132の詳細例に関する実施例であり、図12にそのブロック図を示す。

【0076】図12において、I\_\_ADDR\_\_BUSは命令アドレスバスであり、I\_\_DATA\_\_BUSは命令データバスである。これらのバスを用いて命令メモリ110-1から積和演算命令などの命令が読み出される。またD\_\_ADDR\_\_BUSはデータアドレスバス、D\_\_DATA\_\_BUSはデータバスであり、これらのバスを用いてデータメモリ110-2から第1、第2の積和入力データMDA、MDBなどのデータが読み出される。このように本実施例ではいわゆるハーバードアーキテクチャのバス構成を採用している。

【0077】PA\_\_BUS、PB\_\_BUS、WW\_\_BUS、XA\_\_BUSは内部バスであり、AUX\_\_BUSは制御回路102と積和演算回路104との間でデータのやり取りを行うためのバスである。IA、DAは、各々、制御回路102(CPU)からI\_\_ADDR\_\_BUS、D\_\_ADDR\_\_BUSにアドレスを出力するためのものである。DINは、D\_\_DATA\_\_BUSからのデータを制御回路102に入力するためのものであり、DOUTは、制御回路102からのデータをD\_\_DATA\_\_BUSに出力するためのものである。

【0078】命令デコーダ140は、I\_\_DATA\_\_BUSから入力された命令を受け付けると共に解析し、命令の実行に必要な種々の制御信号を出力する。例えば命令に応じた種々の指示を、即値生成器142を介して制御回路102の各部に与える。また割り込みコントロー

ラ130(図1参照)からの割り込みを受け付けた場合には、割り込みハンドラを起動するTRAP VECTORをD\_ADDR\_BUSに出力すると共に、TRAP信号をアクティブ(=1)にして割り込みが発生したことを積和演算回路104に伝える。また積和演算命令を受け付けた場合には、mac信号をアクティブにして積和演算命令が発行されたことを積和演算回路104に伝える。

【0079】即値生成器142は、命令に含まれる即値に基づき、命令の実行時に使用する32ビットの即値データを生成したり、各命令の実行に必要な0、±1、±2、±4のconstantデータを生成したりする。PCインクリメント118は、1つの命令を実行する毎にPC120の値をインクリメントする処理を行う。アドレス加算器144は、各種レジスタに格納されている情報や即値生成器142で生成される即値データを用いて加算処理を行い、メモリ110からの読み出し処理に必要なアドレスを生成する。

【0080】汎用レジスタ103は16本の32ビットのレジスタR0~R15を含んでいる。SP146は、スタックポインタ専用の32ビットのレジスタであり、スタックの先頭番地を指すスタックポインタを格納する。PSR(プロセッサステータスレジスタ)148は、各種のフラグを格納する32ビットのレジスタであり、本実施例では実行回数のデクリメント処理も行い、ゼロディテクタ134は、ALU132の演算結果が零の場合にALU\_zeroをアクティブ(=1)にする。これによりPSR148にゼロフラグがセットされると共に、実行回数が零になったことが積和演算回路104に伝えられる。バスマルチプレクサ121は、PA\_BUS、PB\_BUS、WW\_BUSのいずれか1つを選択してAUX\_BUSに接続するためのものである。バスマルチプレクサ121はTEMPmレジスタ122、TEMPnレジスタ124を含み、第1、第2の積和入力データMDA、MDBが両方とも揃った時にこれらのデータを積和演算回路104に出力する。

【0081】積和演算回路104は、ステートマシン150を含んでいる。このステートマシン150は、ALU\_zero、trap、macなどの各種の信号に基づいて積和演算回路104の状態を制御する。

【0082】さて図13のタイミングチャートの中のMACステート(MAC0~MAC8)は積和演算回路104(ステートマシン150)の状態を表すものであり、図14(A)にその状態遷移図を示す。ここで状態遷移図の中の各信号の意味は次の通りである。

【0083】 mac

積和演算命令を命令デコーダ140が受け付けた時に1(アクティブ)になる信号である。

【0084】 mac\_end

積和演算命令の終了条件が成立すると1になる信号であり、具体的にはmac\_zero又はmac\_trapが1になると1になる信号である。

【0085】 mac\_zero

積和演算の実行回数が零になった時に1になる信号である。ここで図14(B)に示すように、mac\_zeroは、マイクロコンピュータがリセットされた場合或いはMACステートがMAC8又はMAC9になった場合に0になる。またMACステートがMAC3、MAC5又はMAC7の時にゼロディテクタ134からのALU\_zero信号が1になると1になる。

【0086】 mac\_trap

積和演算命令の実行中に割り込みが発生した場合に1になる信号である。ここで図14(B)に示すように、mac\_trapは、マイクロコンピュータがリセットされた場合或いはMACステートがMAC8又はMAC9になった場合に0になる。またMACステートがMAC5又はMAC7の時に命令デコーダ140からのtrap信号が1になると1になる。

【0087】図14(A)に示すように、積和演算命令が発行されずmac=0の場合には、MACステートはMAC0にとどまる。一方、mac=1になるとMAC1に移行する。MAC1からMAC2、MAC2からMAC3へはクロックに同期して無条件(UCT)に移行する。

【0088】MAC3で、mac\_endが1の場合にはMAC9に移行すると共に、mac\_endが0にリセットされる(図14(B)参照)。MAC9に移行した後、mac=1ならMAC1に戻り、mac=0ならMAC0に戻る。一方、mac\_endが0の場合にはMAC3からMAC4に移行する。

【0089】MAC4からMAC5へはクロックに同期して無条件に移行する。この際、実行回数がデクリメントされるため(図13のE22参照)、mac\_zeroが1になる可能性がある。そこでMAC5で、mac\_endが1か否かを判断し、1の場合にはMAC8に移行し、MAC8からMAC0又はMAC1に戻る。一方、mac\_endが0の場合は、MAC5からMAC6に移行する。

【0090】MAC6からMAC7へはクロックに同期して無条件に移行する。この際、実行回数がデクリメントされるため(図13のE24、E26参照)、mac\_zeroが1になる可能性がある。そこでMAC7で、mac\_endが1か否かを判断し、1の場合にはMAC8に移行し、0の場合はMAC6に戻る。

【0091】例えば積和演算の実行回数が0に設定されていた場合には、MACステートは、まずMAC0、MAC1、MAC2、MAC3と変化する。そしてmac\_end=1(mac\_zero=1)となっているため、MAC3からMAC9、MAC0(又はMAC1)

と変化する。

【0092】実行回数が1に設定されていた場合には、MACステートは、まずMAC0、MAC1、MAC2、MAC3、MAC4と変化する。そしてMAC4からMAC5への移行の際に実行回数がデクリメントされるためmac\_end=1になる。この結果、MACステートはMAC4からMAC5、MAC8、MAC0（又はMAC1）と変化するようになる。

【0093】実行回数が2に設定されていた場合には、MACステートは、MAC0、MAC1、MAC2、MAC3、MAC4、MAC5、MAC6、MAC7、MAC8、MAC0（又はMAC1）と変化する。即ちこの場合には、MAC4からMAC5、MAC6からMAC7の間で実行回数がデクリメントされて零になる。なお実行回数が3以上の場合には、MAC6からMAC7に移行しMAC6に戻る動作を実行回数が零になるまで繰り返すことになる。

【0094】割り込み要求がなされた場合には、MAC5又はMAC7まで状態が進んだ所で初めてmac\_trap=1（mac\_end=1）か否かが判断され、MAC8に移行することになる。

【0095】本実施例のステートマシン150の1つの特徴は、所望の回数の積和演算を完了した場合（mac\_zero=1）又は割り込み要求がなされた場合（mac\_trap=1）にアクティブになるmac\_end信号に基づいて、MACステートを初期ステートMAC0（又はMAC1）に戻す点にある。このようにすることで、所望の回数の積和演算を完了した場合に用いる状態遷移を利用して、割り込み要求がなされた場合に行う状態遷移も実現することが可能となる。これによりステートマシン150の構成の簡素化を図ることができる。

【0096】次に図13を用いて本実施例の動作を説明する。図13は、実行回数が3に設定されている場合のタイミングチャートである。従って、この場合には、MACステートは、MAC0～MAC6、MAC7、MAC6、MAC7、MAC8、MAC0と変化するようになる。また本実施例では汎用レジスタの中のR13が実行回数用のレジスタになっており、実行回数3が設定されている（図13のE0参照）。またR14、R15が第1、第2の積和入力データMDA、MDB用のレジスタになっており、これらのレジスタには、各々、MDA、MDBを格納するメモリ領域の先頭アドレス110h、230hが格納されている（E1、E2参照）。

【0097】図12の命令デコーダ140が積和演算命令を受け付けるとmac=1になり、MACステートがMAC0からMAC1に移行する。

【0098】次に、R13に格納される実行回数がPBバスを介してALU132に出力される（E3）。ALU132は実行回数に0を加算する（E4）。ここで0

を加算するのは、最初に設定された実行回数が0か否かを調べるためである。0の場合には、ALU\_zeroが1になり積和演算命令の実行が終了する（図14

（A）のMAC3、MAC9参照）。

【0099】次に、R14に格納されるアドレス110hがXA\_BUSを介してD\_ADDR\_BUSに出力される（E5、E6）。そしてこのアドレスに基づき第1の積和入力データMDA（110h）がメモリ110から読み出される（E7）。同様にR15に格納されるアドレス230hがXA\_BUSを介してD\_ADDR\_BUSに出力され（E8、E9）、このアドレスに基づき第2の積和入力データMDB（230h）がメモリ110から読み出される（E10）。そしてこれらのMDA、MDBを乗算器105が乗算し（E11）、乗算結果を加算器106-1が加算し（E12）、加算結果をALR107-1に格納する（E13）。そして加算によりキャリー又はボローが生じた場合には、インクリメント・デクリメント106-2がインクリメント又はデクリメント処理を行い（E14）、その結果をAHR107-2に格納する（E15）。

【0100】R14、R15に格納されるアドレス110h、230hはXA\_BUSを介してアドレス加算器144にも出力される（E5、E8）。アドレス加算器144は、これらのアドレスに+2を加算し（E16、E17）、加算結果をWW\_BUSを介してレジスタR14、R15に戻す（E18、E19）。これによりR14、R15に格納されるアドレスが112h、232hに変化し（E20、E21）、次の積和入力データMDA（112h）、MDB（232h）を読み出すことが可能になる。

【0101】ALU132は、MAC4において実行回数を3から2にデクリメントする（E22）。そしてデクリメントされた実行回数がPB\_BUSに出力され、PB\_BUSからALU132の入力に戻される（E23）。次にALU132は、実行回数を2から1にデクリメントする（E24）。そしてデクリメントされた実行回数がALU132の入力に戻される（E25）。次にALU132は、実行回数を1から0にデクリメントする（E26）。すると実行回数が0になったのでALU\_zeroが1になる（E27）。するとMACステートがMAC6からMAC7、MAC8、MAC0と変化する（E28）、積和演算命令の実行が終了する。この際、デクリメントされて0になった実行回数はWW\_BUSを介してR13に格納されることになる（E29、E30）。

【0102】次に図15のタイミングチャートを用いて、割り込みが発生した場合の本実施例の動作について説明する。図15に示すように、例えばMACステートがMAC3の時に割り込みが発生しtrap信号が1になった場合（図15のF1）を考える。この場合、本実

施例では、次のMACステートであるMAC 4では、割り込みがなかった場合と同様の処理が行われる。そしてMACステートがMAC 5になった時に初めて、MAC 5からMAC 8、MAC 0と変化する処理を行う（F 2）。

【0103】即ち本実施例では、レジスタR 14、R 15の内容が、割り込み処理終了後の積和演算の継続実行の際に使用される内容に変化した後に（F 3、F 4、F 5、F 6）、MACステートが初期ステートMAC 0（又はMAC 1）に戻る。このようにすることで、割り込み処理の終了後に、112h、232hのアドレスにある積和入力データMDA、MDBに基づいて積和演算を適切に継続実行することが可能となる。

【0104】また本実施例では、積和演算の実行回数がデクリメントした後に（F 7）、MACステートが初期ステートに戻る。従って、R 13には、デクリメント後の実行回数2が格納されることになり（F 8、F 9）、割り込み処理の終了後に、残りの2回の積和演算を継続実行することが可能となる。

【0105】以上のように処理することで、前述の実施例1、2、4で説明した種々の処理を実現できる。なお実施例3の処理を実現するためには、積和入力データMDA、MDBのメモリからの読み出しを、1回のメモリアクセス（1クロック）で行うようにすればよい。

【0106】（実施例6）実施例6は、本発明が適用されるマイクロコンピュータの詳細例について説明する実施例である。

【0107】図16に示すように実施例6のマイクロコンピュータ700は、32ビットマイクロコンピュータであり、CPU（制御回路、積和演算回路、ALU）710、ROM720、RAM730、高周波発振回路910、低周波発振回路920、リセット回路930、プリスケラ940、16ビットプログラマブルタイマ950や8ビットプログラマブルタイマ960やクロックタイマ970などのタイマ回路、インテリジェントDMA980や高速DMA990などのデータ転送制御回路、割り込みコントローラ800、シリアルインターフェース810、BCU（バスコントロールユニット）740、A/D変換器830やD/A変換器840などのアナログインターフェース回路、入力ポート850や出力ポート860やI/Oポート870などのI/O回路、及びそれらを接続する各種バス750、760、各種ピン890を含む。

【0108】1チップの半導体基板上に形成されるこのマイクロコンピュータ700は、32ビットのデータを処理できるRISC方式のマイクロコンピュータである。そしてパイプライン方式及びロード・ストア方式のアーキテクチャを採用し、ほとんど全ての命令を1クロックの期間で実行する。全ての命令は16ビットの固定長で記述されており、これにより極めて小さい命令コ

ードサイズを実現している。

【0109】そして、実施例1〜5で説明したように、CPU710は、1つの積和演算命令で複数回の積和演算を実行できるようになっている。このため、このマイクロコンピュータ700は、これまでDSP、画像処理専用IC、音処理専用ICなどが行っていた処理を代行することができ、このマイクロコンピュータ700が組み込まれる電子機器の低コスト化、小型化を図ることが可能となる。

【0110】（実施例7）実施例7は、実施例1〜6で説明したマイクロコンピュータを含む電子機器に関する実施例である。

【0111】例えば図17（A）に電子機器の1つであるカーナビゲーションシステムの内部ブロック図を示し、図18（A）にその外観図を示す。カーナビゲーションシステムの操作はリモコン510を用いて行われ、GPSやジャイロからの情報に基づいて位置検出部520が車の位置を検出する。地図などの情報はCDROM530（情報記憶媒体）に格納されている。画像メモリ540は画像処理の際の作業領域になるメモリであり、生成された画像は画像出力部550を用いてドライバーに表示される。マイクロコンピュータ500は、リモコン510、位置検出部520、CDROM530などのデータ入力源からデータを入力し、種々の処理を行い、処理後のデータを画像出力部550などの出力装置を用いて出力する。

【0112】これまでのカーナビゲーションシステムでは、画像処理（グラフィック処理）は、DSPや専用の画像処理ICが行っていた。このため、例えばCISC型のマイクロコンピュータとDSPというように電子機器内に2つのプロセッサが存在することになり、システムが複雑化していた。実施例1〜6で説明したマイクロコンピュータを採用すれば、複数回の積和演算命令の実行を効率よく行うことができるため、DSP等を用いることなく、カーナビゲーションシステムが必要とする画像処理を実現することが可能となる。

【0113】図17（B）に電子機器の1つであるゲーム装置の内部ブロック図を示し、図18（B）にその外観図を示す。このゲーム装置では、ゲームコントローラ560からのプレーヤの操作情報、CDROM570からのゲームプログラム、ICカード580からのプレーヤ情報等に基づいて、画像メモリ590を作業領域としてゲーム画像やゲーム音を生成し、画像出力部610、音出力部600を用いて出力する。マイクロコンピュータ500は、実施例1〜6で説明した積和演算機能を用いて、座標変換、透視変換、クリッピングなどの3次元画像処理や、音圧縮、音伸長などの音処理を行うことになる。

【0114】図17（C）に電子機器の1つであるプリンタの内部ブロック図を示し、図18（C）にその外観

図を示す。このプリンタでは、操作パネル620からの操作情報、コードメモリ630及びフォントメモリ640から文字情報に基づいて、ビットマップメモリ650を作業領域として、印刷画像を生成し、プリント出力部660を用いて出力する。またプリンタの状態やモードを表示パネル670を用いてユーザに伝える。マイクロコンピュータ500は、実施例1～6で説明した積和演算機能を用いて、直線や円弧の描画、画像の拡大、縮小などの処理を行うことになる。

【0115】なお本発明のマイクロコンピュータを適用できる電子機器としては、上記以外にも例えば、携帯電話（セルラーフォン）、PHS、ページャ、オーディオ機器、電子手帳、電子卓上計算機、POS端末、タッチパネルを備えた装置、プロジェクタ、ワードプロセッサ、パーソナルコンピュータ、テレビ、ビューファインダ型又はモニタ直視型のビデオテープレコーダなど種々のものを考えることができる。

【0116】なお、本発明は上記実施例1～7に限定されるものではなく、本発明の要旨の範囲内で種々の変形実施が可能である。

【0117】例えば積和演算命令による積和演算の実行回数の特定手法は、上記実施例で説明したものに限らず、種々の変形実施が可能である。

【0118】また本発明の情報処理回路は、マイクロコンピュータ、特にRISC型のマイクロコンピュータに適用した場合に特に有効であるが、それ以外の用途も可能である。

【0119】また積和演算命令の記述構成も本実施例で説明したものに限られるものではなく、種々の変形実施が可能である。

【0120】

【図面の簡単な説明】

【図1】マイクロコンピュータの構成例を示すブロック図である。

【図2】実施例1の動作を説明するためのフローチャートである。

【図3】図3（A）、（B）、（C）は、レジスタとメモリに格納されるデータの関係について説明するための図である。

【図4】比較例の動作を説明するためのフローチャートである。

【図5】図5（A）、（B）は、比較例の問題点について説明するための図である。

【図6】図6（A）、（B）、（C）は、積和演算命令の種々の実施形態について説明するための図である。

【図7】実施例2の動作を説明するためのフローチャートである。

【図8】図8（A）、（B）は、レジスタとメモリに格納されるデータの関係について説明するための図である。

【図9】実施例3の動作を説明するためのフローチャートである。

【図10】図10（A）は、実施例3のメモリへのデータ格納手法について説明するための図であり、図10（B）は比較例のタイミングチャートの例であり、図10（C）は、実施例3のタイミングチャートの例である。

【図11】図11（A）は実施例4の構成例を示すブロック図であり、図11（B）はそのタイミングチャートの例である。

【図12】実施例5の構成例を示すブロック図である。

【図13】実施例5のタイミングチャートの例である。

【図14】図14（A）、（B）は、ステートマシンについて説明するための図である。

【図15】割り込み発生時のタイミングチャートの例である。

【図16】実施例6のマイクロコンピュータの構成例である。

【図17】図17（A）、（B）、（C）は、種々の電子機器の内部ブロック図の例である。

【図18】図18（A）、（B）、（C）は、種々の電子機器の外観図の例である。

【符号の説明】

101 マイクロコンピュータ

30 102 制御回路

103 汎用レジスタ

104 積和演算回路

105 乗算器

106 加算器

107 MACレジスタ（積和結果用レジスタ）

108 BCU（バスコントロールユニット）

109 内部データバス

110 メモリ

111 外部アドレスバス

40 112 外部データバス

120 PC（プログラムカウンタ）

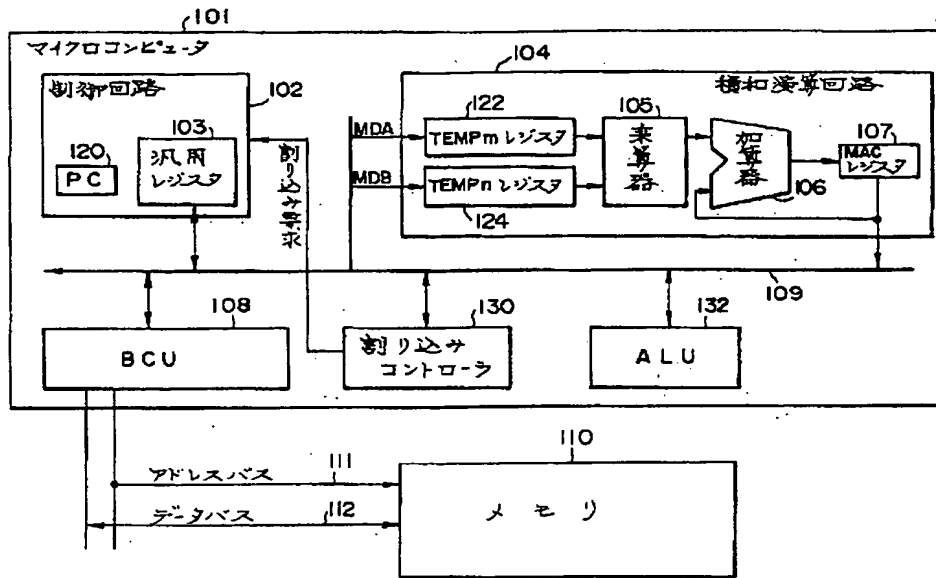
122 TEMPmレジスタ

124 TEMPnレジスタ

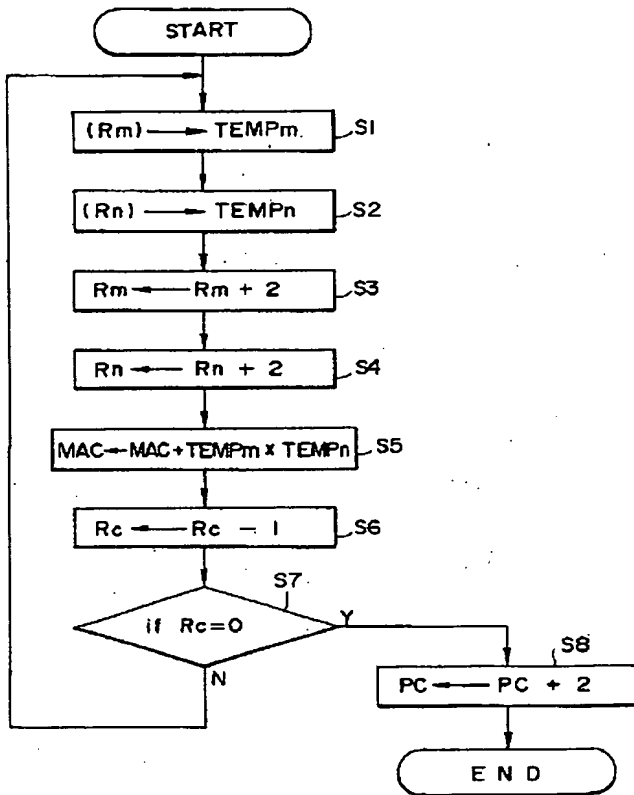
130 割り込みコントローラ

132 ALU

【图 1】



【圖 2】



【図4】

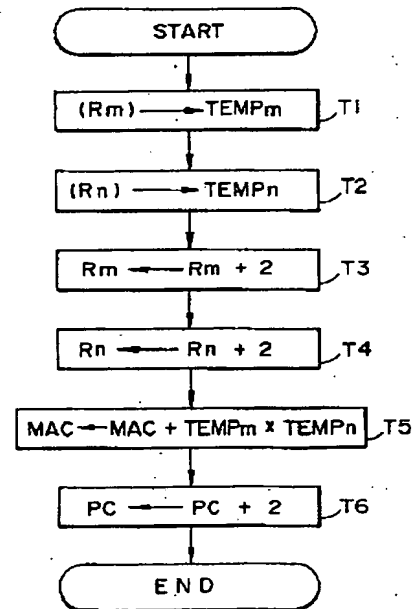




Figure 1 consists of three schematic diagrams labeled (A), (B), and (C), illustrating different data transfer methods between a 110-bit register and a 120-bit register.

**(A) First Embodiment:** A 110-bit register (MDBL, ..., MDB0) is connected to a 10-bit register (MDA1, ..., MDA0). The 10-bit register is connected to a 120-bit register (Rn, ..., Rc=0). The 120-bit register is connected to a PC (120) and a 12-bit register (MDBL, ..., MDB0).

**(B) Second Embodiment:** A 110-bit register (MDBL, ..., MDB0) is connected to a 10-bit register (MDA1, ..., MDA0). The 10-bit register is connected to a 120-bit register (Rn, ..., Rc=Rc-1). The 120-bit register is connected to a PC (120) and a 12-bit register (MDBL, ..., MDB0).

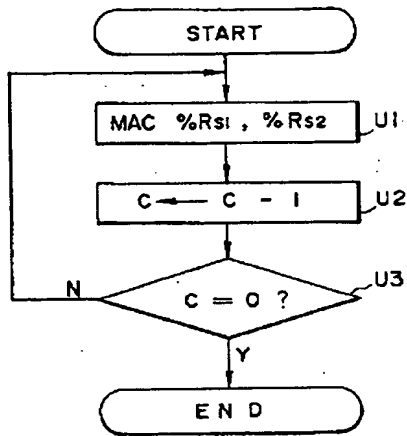
**(C) Third Embodiment:** A 110-bit register (MDBL, ..., MDB0) is connected to a 10-bit register (MDA1, ..., MDA0). The 10-bit register is connected to a 120-bit register (Rn, ..., Rc=0). The 120-bit register is connected to a PC (120) and a 12-bit register (MDBL, ..., MDB0).

【図5】

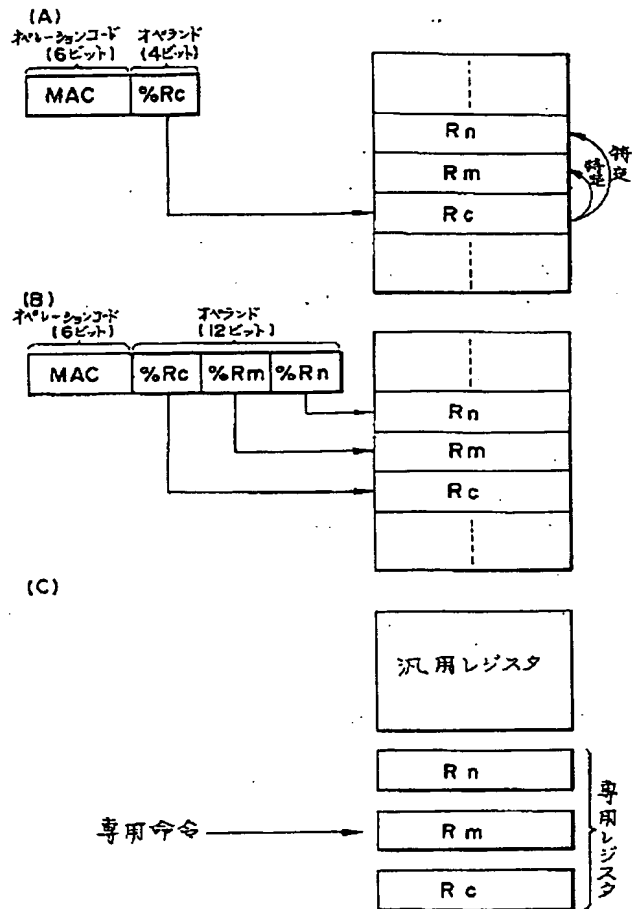
(A)

MAC %Rs1, %Rs2  
 MAC %Rs1, %Rs2  
 MAC %Rs1, %Rs2  
 MAC %Rs1, %Rs2  
 ⋮

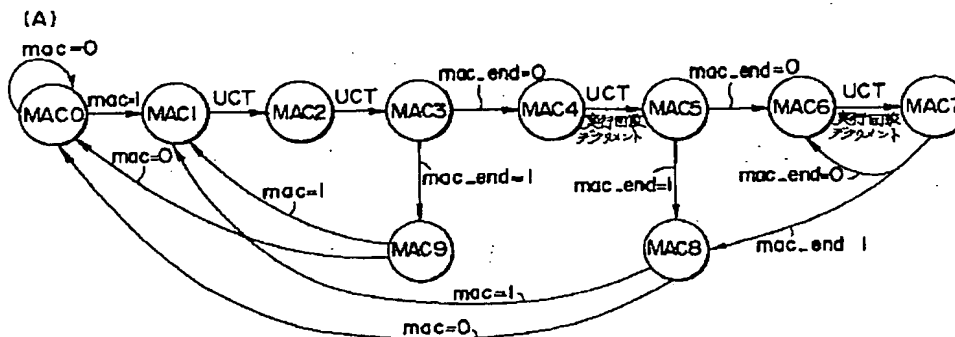
(B)



【図6】



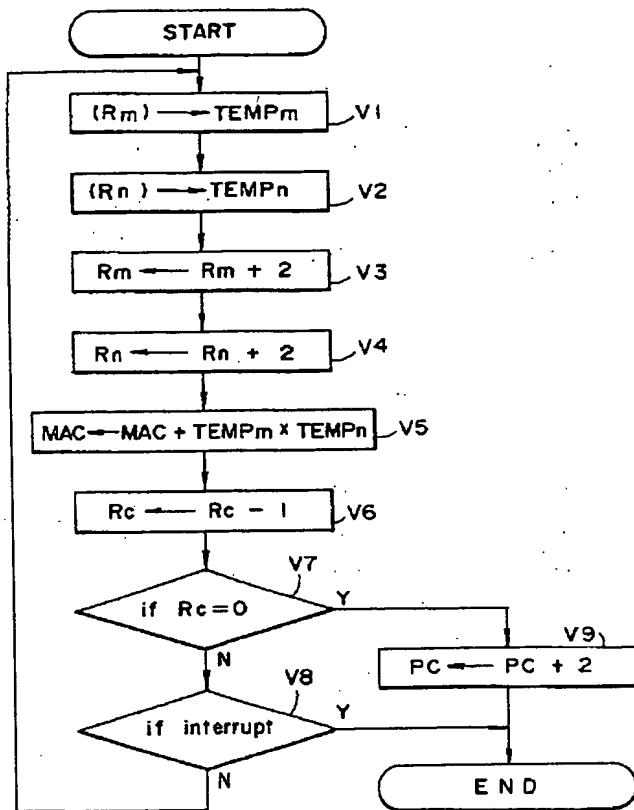
【図14】



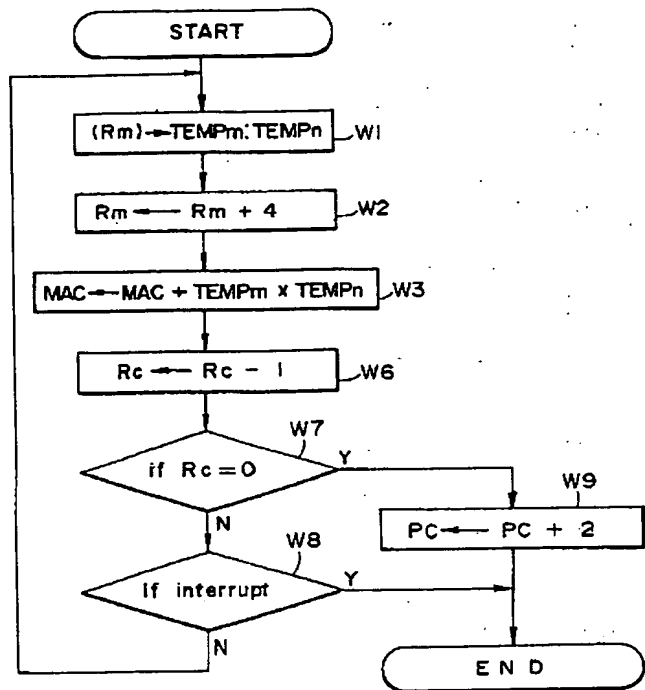
(B)

		インアクティブ (=0) 条件	アクティブ (=1) 条件
mac_end	mac_zero	リセット or MAC8 or MAC9	ALU_zero AND (MAC3 or MAC5 or MAC7)
(mac_zero or mac_trap)	mac_trap	リセット or MAC8 or MAC9	trap AND (MAC5 or MAC7)

【図7】

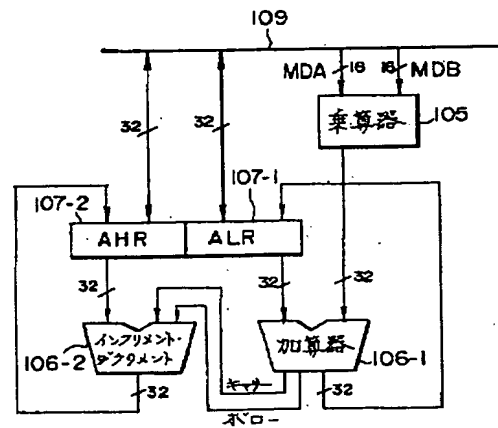


【図9】

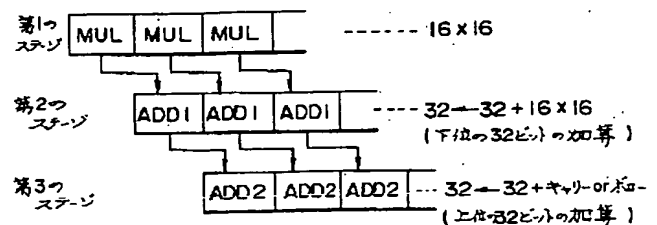


【図11】

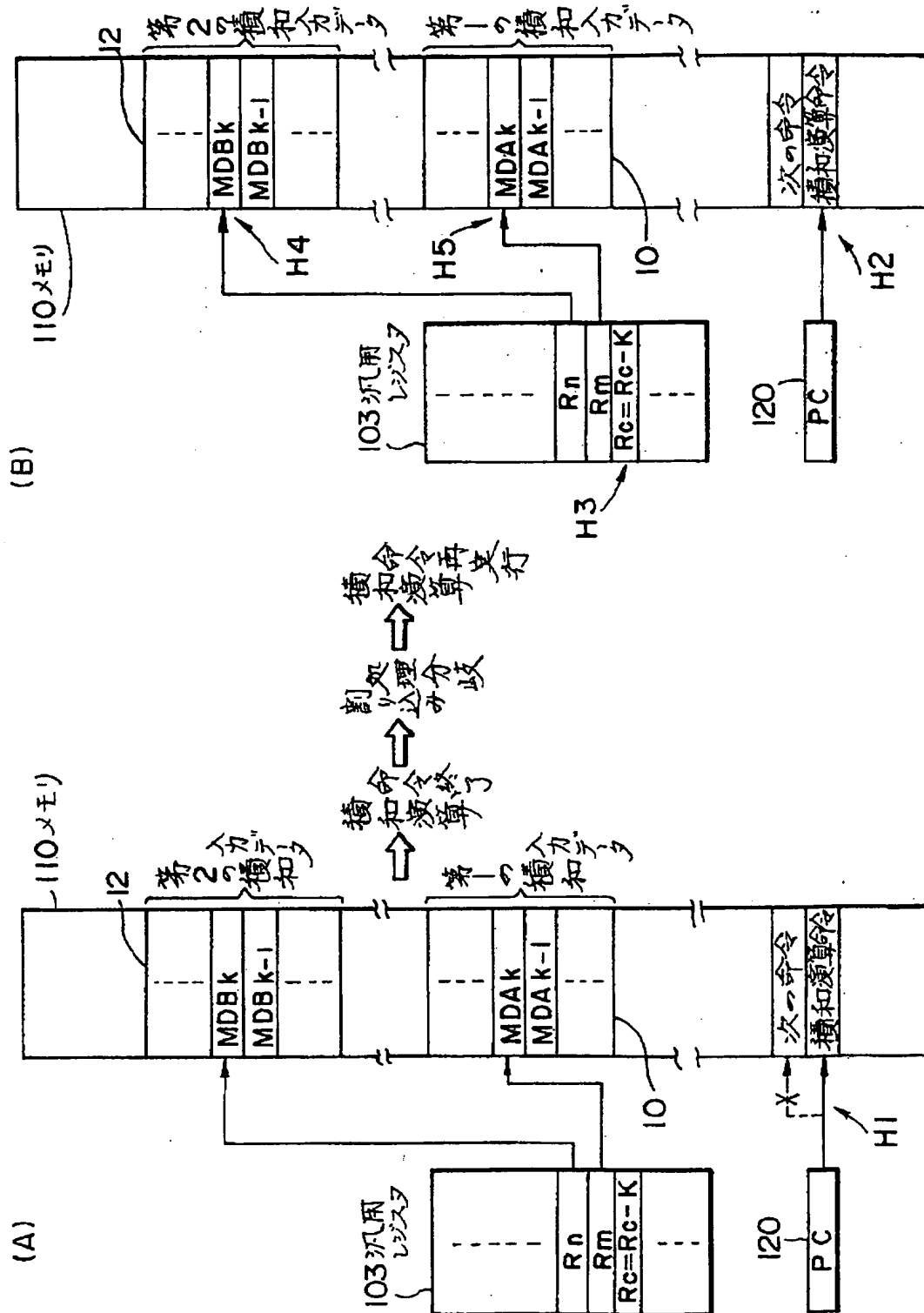
(A)



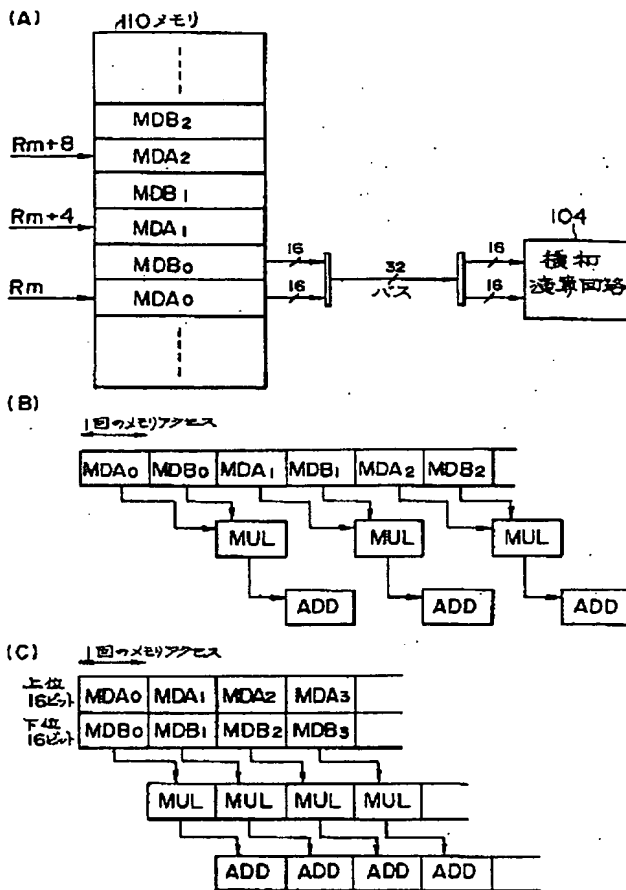
(B)



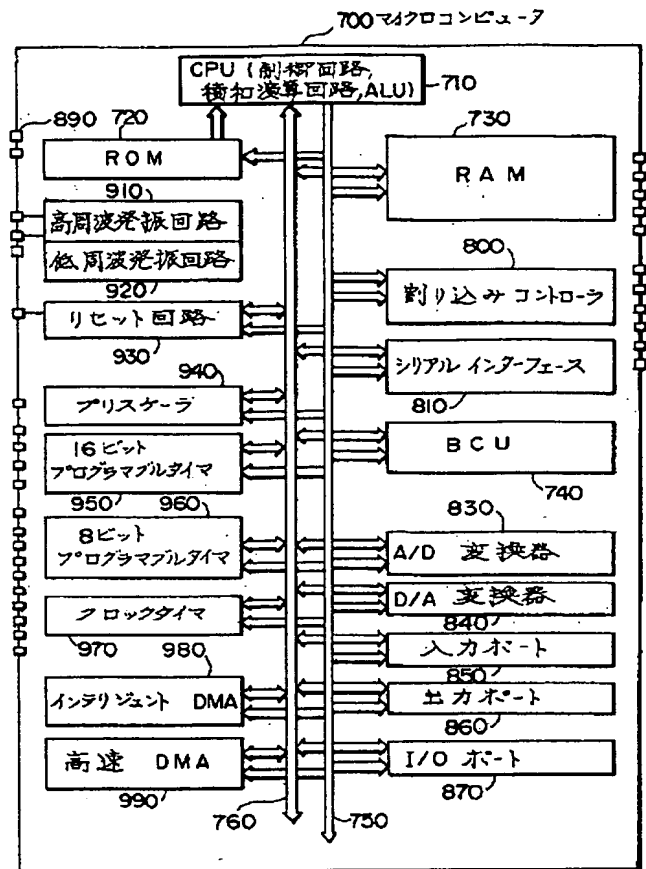
【図8】



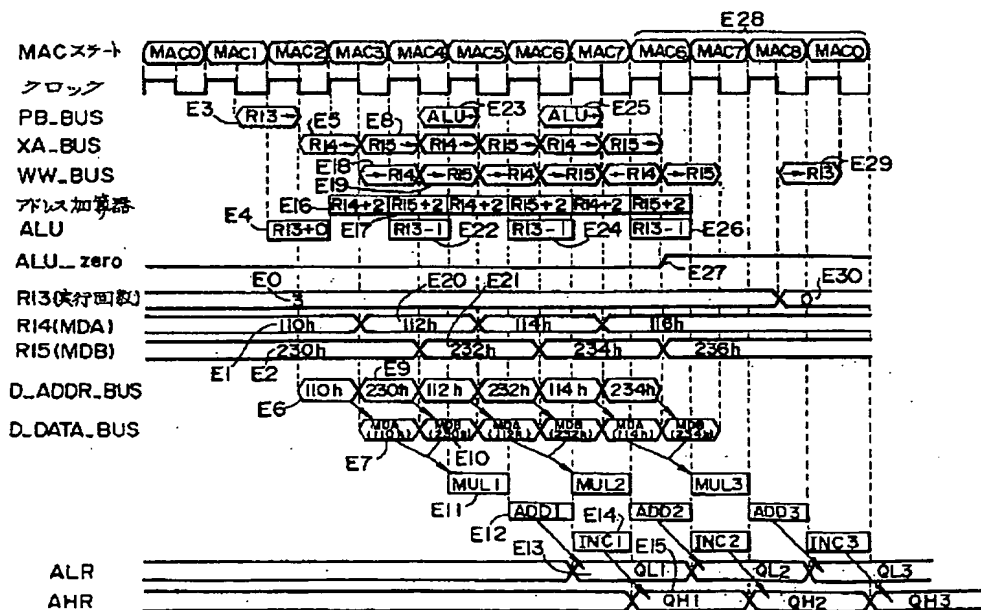
【図10】



【図16】

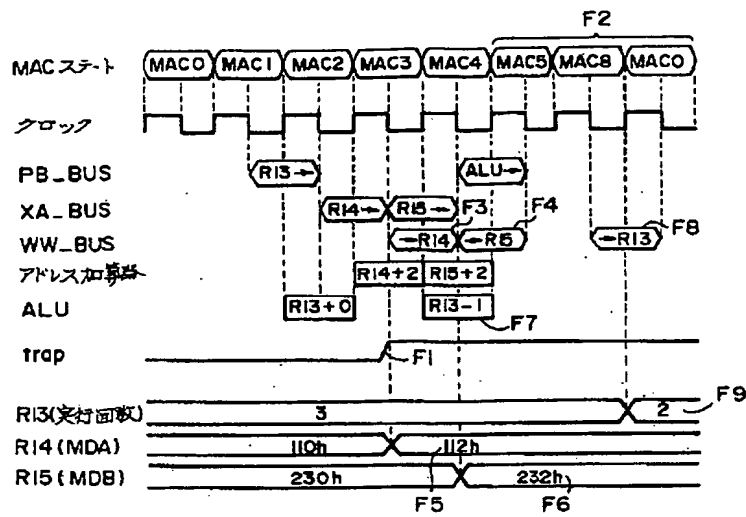


【図13】

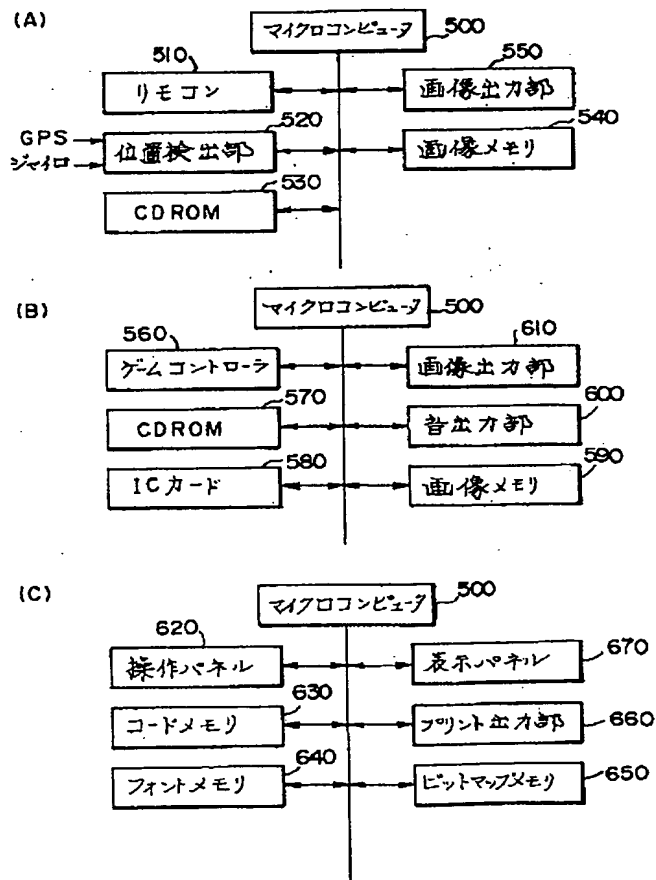


[illegible]

【図15】



【図17】



【図18】

